

## Article

# Control of Hydraulic Pulse System Based on the PLC and State Machine Programming <sup>†</sup>

Juraj Pančík <sup>1,\*</sup> and Pavel Maxera <sup>2</sup> 

<sup>1</sup> Institute of Forensic Engineering, Brno University of Technology, Purkyňova, 464/118, Brno 61200, Czech Republic

<sup>2</sup> Institute of Forensic Engineering, Brno University of Technology, Purkyňova, 464/118, Brno 61200, Czech Republic; pavel.maxera@usi.vutbr.cz

\* Correspondence: juraj.pancik@gmail.com; Tel.: +421-908-963-289

<sup>†</sup> This paper is an extended version of paper published in the International Conference Applied Electronics 2015, University of West Bohemia, Pilsen, Czech Republic, pp. 175–180, ISBN 978-80-261-0386-8.

Received: 17 September 2018; Accepted: 14 November 2018; Published: 20 November 2018



**Abstract:** In this paper, we deal with a simple embedded electronic system for an industrial pneumatic–hydraulic system, based on a low-cost programmable logic controller (PLC) and industrial electronic parts with 24 V logic. The developed system is a hydraulic pulse system and generates a series of high-pressure hydraulic pulses with up to a max. 200 bar output pressure level and with up to a max. 2 Hz output hydraulic pulses frequency. In this paper we are describing requirements, the concept of the embedded control system in a diagram, security features and its industrial network connectivity (CAN bus, MODBUS). In description of the software solution we describe the implementation of the program threads approach in this low-cost PLC. The PLC programming with threads generate two layers of services—physical and application layer, and as a result, the threads create the main control state machine. In conclusion, we describe the calibration method of the system and the calibration curves. For further study we offer readers the full programming code written in sequential function charts to be used as PLC language. The cost of the described industrial networked control system with industry standard optoelectronic insulated interfaces and certified industrial safety relay does not exceed €1000 Euros.

**Keywords:** PLC programming; hydraulic pulse system; state machine programming

## 1. Introduction

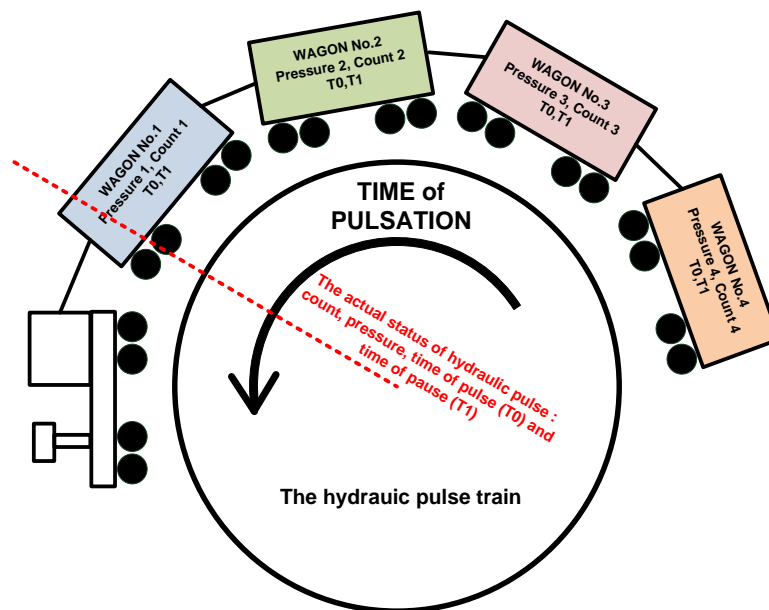
Our built-in embedded system for the pneumatic–hydraulic pulse system is based on the use of a modern low-cost programmable logic controller (PLC). A modern PLC allows you to program parallel running programs (threads), each of them being programmed for PLC programmers well-known ladder diagrams or as functional block diagrams [1,2]. In our work, we prepared for the PLC robust, external error-free control program, so as to control the series high pressure hydraulic pulses based at one PLC. Each thread represents a separate task and independent state of system. Parallel running threads cooperate with each other and this set of independent threads creates a unique reactive control state machine [3,4].

## 2. Requirements for Pneumatic–Hydraulic Pulse System

Since we have limited space in this contribution, we are only trying to describe the requirements for the overall machine and requirements for creating a specific series of impulses.

The system controls up to four independent series, each of them defined by four adjustable parameters; the number of pulses (N), the maximum hydraulic pressure (p), the hydraulic pulse

duration ( $T_1$ ) and the duration of the pause between pulses ( $T_0$ ). Each WAGON of this TRAIN has four passengers:  $N$ ,  $p$ ,  $T_1$ ,  $T_2$ , see Figure 1, where WAGONS build a so-called “PNEUMATIC–HYDRAULIC PULSE TRAIN”. The pneumatic–hydraulic pulse train can cyclically repeat its defined circle path by the number of times which is set in the TRAIN SESSION number. The number of WAGONS may be set by the user from 1 to 4. The embedded control system allows each user to set the output of the hydraulic pressure in bars within the range 0–200 bar, the number of WAGONS, and number of cyclic reruns of the train’s path. The control system must allow system calibration (dependence of the output hydraulic pressure in units of [bar] at the user’s desired hydraulic pressure in [ADU] units (ADU unit = Analog Digital Unit)). It must ensure the generation of system and error messages.



**Figure 1.** Pneumatic–hydraulic pulse train-schematic represents of cyclic repeating sequence pulses for hydraulic pulse system as cyclic paths of train with four wagons, each with four passengers.

### 3. Hardware Solution

#### 3.1. Pneumatic–Hydraulic Parts

The schematic diagram of the developed pneumatic–hydraulic system with PLC based control system is shown in Figure 2 and a photo of the prototype is shown in Figure 3. The proportional pneumatic valve regulates input air pressure. Regulated air is then fed into pneumatic four-way valve circuit and it controls the two-position pneumatic pump. This pneumatic circuit controls the two-direction movement of the hydraulic cylinder with, resulting amplified hydraulic pressure as its output [5].

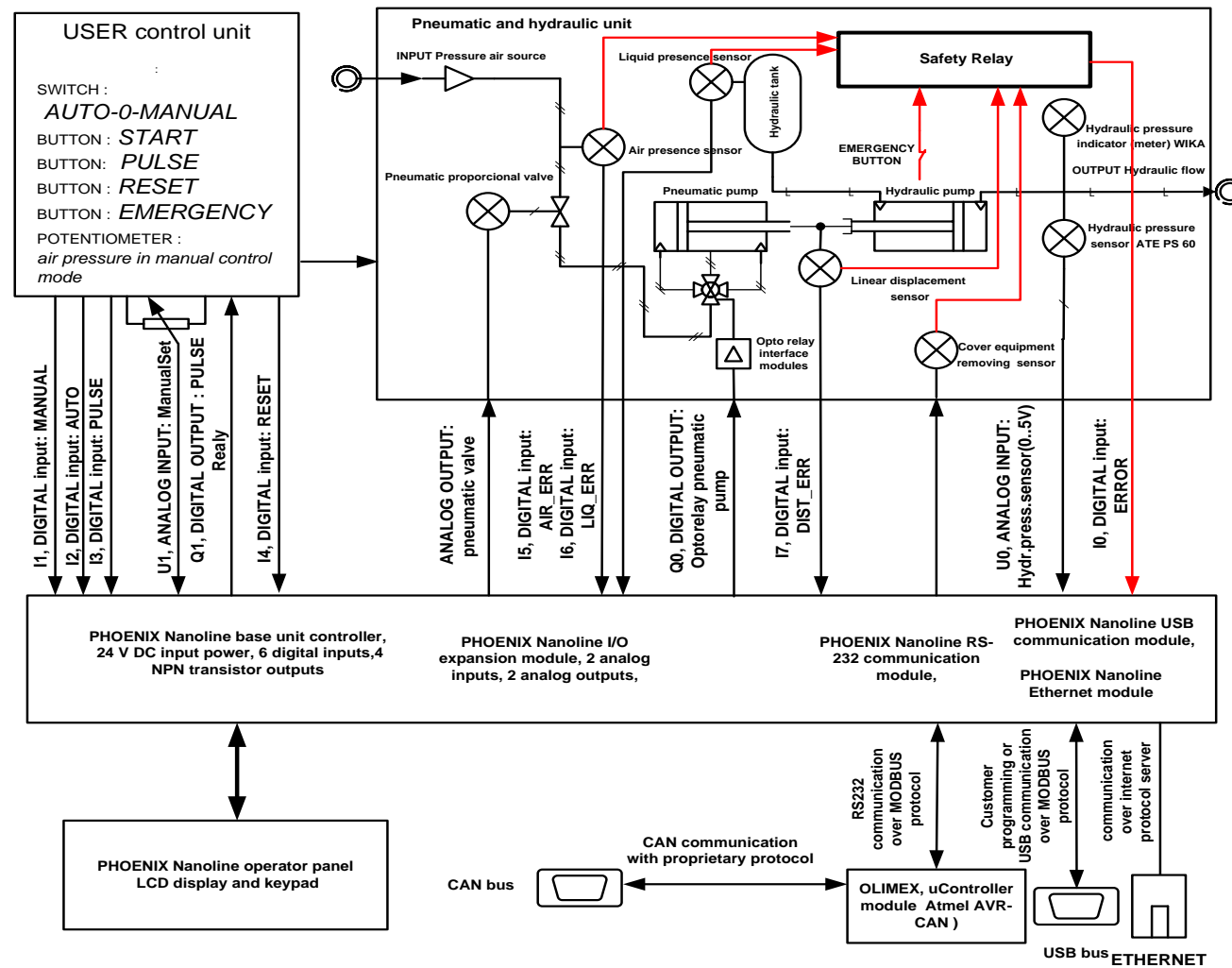
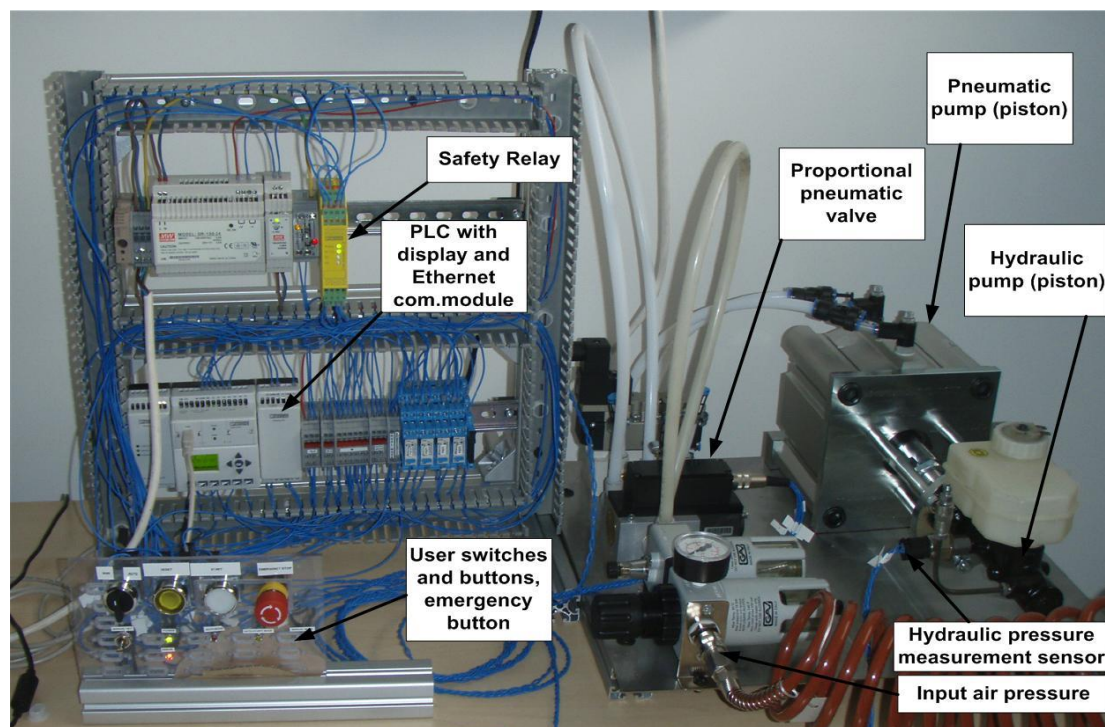


Figure 2. Schematic diagram of the developed pneumatic-hydraulic system.



**Figure 3.** The photo of the functional prototype of the pneumatic–hydraulic pulse system.

### 3.2. PLC Based Control System

The embedded control system was implemented on a cheap programmable logic controller type (PLC) Nanoline from Phoenix Contact [6] and on additional PLC's modules (analog input–output module, LCD display and keypad user interface module, USB, RS232 and Ethernet communication module). The PLC works with industrial logic levels and they both have two levels 0 and 24 V; PLC's analog outputs are in the range 0–10 V, and PLC's analog inputs are in range 0–5 V. The used PLC Nanoline is built around an ARM 32 processor and its operating system allows for the running of several independent program threads simultaneously. The run of each thread is conditional on the value of the hardware input signal or by the program setting of value of the internal program flag. This interesting ability of the PLC's operating system creates a challenge for us to generate a master program for control of hydraulic pulses based on states machines

### 3.3. User Interface Elements

HMI (human–machine interface) of the pneumatic–hydraulic pulse system consists of keypad, LCD display (4 rows  $\times$  20 characters) and few switches. The manual switch strains between two basic modes of equipment (MANUAL mode and AUTO mode), the potentiometer serves for manual adjustment of output hydraulic pressure, the button allows for starting the series hydraulic pulsation (PULSE mode), another button serves to stop the pulses (the RESET button), the security (or emergency) button serves to turn off the power of the appliance (EMERGENCY). The required values for WAGON and TRAIN parameters and the choosing of other equipment modes (e.g., calibration of the system) are entered from the PLC build keypad.

### 3.4. Electrical Parts of System

Electrical parts of system consist a 24 V power supply, a safety relay, electrical and optical relays, switches, terminals and wires as is seen in Figures 2 and 3.

### 3.5. Safety Measures

The basis of ensuring the safety operation of high pressure pneumatic–hydraulic pulse system is the certified safety relay [7]. The safety relay can detect system errors that switch off the power for the whole electrical power part of the equipment (but not for control PLC). Key safety sensors are connected in the series. In the case of detection of problems by dedicated sensors with any media (hardware signals AIR or LIQUID) or detection of mechanical problems (hardware signals as DISTANCE ERROR or COVER EMERGENCY SWITCH), the series of connected sensor's contacts is interrupted, the safety relay responds and sets the signal ERROR and switches off the power for the hydraulic–pneumatic pulse system in order to automatically reduce the output hydraulic pressure to minimum. The mechanical error with name DISTANCE ERROR is generated by a mechanical contact switch mounted near the hydraulic cylinder and serves to indicate an attempt to generate high hydraulic pressure. To ensure of the safe operation of the high pressure pneumatic–hydraulic pulse system, we used a certified safety relay. The safety relay can, in the case of detecting the system errors, switch off the power for the whole electrical power part of the equipment (except of power for control PLC).

### 3.6. Network Connectivity of the Pneumatic–Hydraulic System

Network connectivity of our control system is ensured with the PLC expansion Ethernet communication module [8]. This expansion module allows for external control of all program elements in the PLC program (e.g., PLC's signals, flags, registers ...) via industrial standard MODBUS protocol (via TCP-IP protocols). For our special purpose, where the pneumatic–hydraulic pulse system is a part of a test bench, we have developed the communication embedded server—a converter between the CAN bus (CAN is used by test bench's protocol) and RS232 (as physical base for MODBUS protocol). The CAN converter consists of the hardware module OLIMEX occupied with 8bit ATMEL AVR-CAN microcontroller [9]. The developed protocol and BUS converter allows us to control and monitor pneumatic–hydraulic pulses system, even though the CAN bus.

## 4. Software Solution

### 4.1. Basic PLC Functions

The programming approaches for modern PLC are described in standards and books, for example by the authors of [10,11]. The ladder diagram or graphic function block diagram programming approach is still the focus of PLC's programmers. The IEC 61131-3 (IEC 1131-3: The International Programmable Controller Language Standard) is the third part (of 10) of the open international standard IEC 61131 for programmable logic controllers and was first published in December 1993 by the IEC [6]. The current (third) edition was published in February 2013. Part 3 of IEC 61131 deals with the basic software architecture and programming languages of the control program within the PLC. It defines two graphical and two textual programming language standards:

- Ladder diagram (LD), graphical
- Function block diagram (FBD), graphical
- Structured text (ST), textual
- Instruction list (IL), textual (deprecated in 3rd edition of the standard)
- Sequential function chart (SFC), has elements to organize programs for sequential and parallel control processing (IEC 1131-3: The International Programmable Controller Language Standard and)

The modern low-cost PLC based on powerful ARM architecture such as Nanoline from Phoenix Contact allows for the running of several program threads. It gives to the PLC programmer the possibility to program the main control program without creating one long and complicated PLC program. Each independent PLC program thread can be initiated by hardware signal(s) (the PLC use



24 V logic at input) or by internal programming flag with boolean data type (it can be set by another PLC program thread). Each PLC program thread can be programmed with ladder diagram or graphic functional block diagram programming approach and the PLC programmer may not know about the concurrent program techniques or about the use of special concurrent program library for PLC.

#### 4.2. List of Hardware Inputs and Outputs for PLC

The list of all digital and analog inputs and outputs used in the project is in the Table 1. Each hardware input signal (ERROR, WAIT, MANUAL, AUTO, PULSE and RESET) is processed in the PLC by individual program thread (see Table 2). These signals are grouped in the so-called physical layer. Some hardware signal inputs are used for indicating the error's cause in the safety power shutdown of the pneumatic-hydraulic system via safety relay (Figure 2).

**Table 1.** List of programmable logic controller (PLC) hardware inputs and outputs and their responsible processing layer.

n.	Name of Hardware Signal	Signal Direction/Digital/Analog	Signal is Processed in the Independent Thread	Description of Physical Signal
1	ERROR	I0-input	Physical layer 4_ERROR	From safety relay
2	MANUAL	I1-input	Physical layer 1_MANUAL	User input, manual control
3	AUTO	I2-input	Physical layer 2_AUTO	User input, pulse gen.
4	PULSE	I3-input	Physical layer 3_PULSE	User input, start of pulsing
5	RESET	I4-input	Physical layer 5_RESET	User input, stop of pulsing
6	AIR_ERROR	I5-input	Application layer 4_State_Func	From pressure air tank switch
7	LIQUID_ERROR	I6-input	Application layer 4_State_Func	From hydraulic tank switch
8	DISTANCE_ERROR	I7-input	Application layer 4_State_Func	From distance contact switch
9	PNEU_PUMP	Q0-output	Application layer 1_State_Func and 3_State_Func	For pneumatic pump control
10	PULSE_RELAY	Q1-output	Application layer 3_State_Func	For pulse control generation
11	HYDR_PRES_SENSOR	AI0-input	All application layers	From hydraulic pressure sensor
12	MANUAL_PRES_SET	AI1-input	Application layer 1_State_Func	From user man. potentiometer
13	PNEUMATIC_VALVE	AU0-output	Application layer 1_State_Func and 3_State_Func	For proportional valve control

**Table 2.** List of implemented physical layer threads.

n.	Name of Thread	Signal Condition for Perform of the Thread	Program Switches the Flag to ON	Conditionally can Set also	Description of Program Thread
1	4_ERROR	Signal ERROR = ON	State_FLAGS_4_ERROR	PNEU_PUMP(OFF) PULSE_RELAY(OFF)	Safety relay output processing
2	0_WAIT	Signal ERROR = ON	State_FLAGS_0_WAIT		Wait state
3	1_MANUAL	Signal MANUAL = ON AND ERROR = OFF	State_FLAGS_1_MAN	State_FLAGS_0_WAIT	User input, manual control mode
4	2_AUTO	Signal AUTO = ON AND ERROR = OFF	State_FLAGS_2_AUTO	State_FLAGS_0_WAIT	User input, pulse generation mode
5	3_PULSE	Signal PULSE = ON AND ERROR = OFF	State_FLAGS_3_PULSE	State_FLAGS_0_WAIT	User input, start of pulsing
6	5_RESET	Signal RESET = ON AND ERROR = OFF	State_FLAGS_2_AUTO	START_BUTT_ON	User input, stop of pulsing

#### 4.3. Physical Layer Threads

Threads of the physical layer (Table 2) run in the PLC's operating system after their initializing by hardware input signals. The physical layer ensures processing of hardware signals and as a result, this layer sets outputs flags, which are processed in the application layer. These output flags manage transitions between states in the master control state machine (literally by start or by stop of the respective threads in the application layer). These program threads also process aforementioned inputs hardware signals and set of hardware output signals by aligning the required settling time or also, the next set of parameters of elements of pneumatic or hydraulic hardware (e.g., delay of relays, setting time of input valve, etc.).

#### 4.4. Application Layer Threads

The running of the application layer thread (listed in Table 3) begin by changes of program flags (one or two flags, then in AND or in OR mutual combination) which are received from the physical layer. Some threads in application layer create of individual states in the master control state diagram (Figure 3). In addition, these threads enable the realization of sub states. For example, thread 2\_State\_Func can call of sub state 6\_State\_Func. This thread serves for the manual input of parameters in the hydraulic pulse train. Next threads also underpin subroutine for user input interaction such the 22\_State\_Func. This sub state provides the possibility of the manual input of calibration constants.

**Table 3.** List of application layer threads.

n.	Name of Thread	Flag Condition for Perform of the Thread	Description of Program Thread, see Figure 4 Master Control State Diagram for Hydraulic Pulse System
1	0_State_Func	State_FLAGS_0_WAIT = ON AND ERROR = OFF	WAIT STATE, Initialization of calibration constants, zeroing of the ATE pressure sensor
2	1_State_Func	State_FLAGS_1_MANUAL = ON AND ERROR = OFF	MANUAL STATE, manual control of output hydraulic pressure via manual potentiometer
3	2_State_Func	State_FLAGS_2_AUTO = ON AND ERROR = OFF	AUTO STATE, setting of parameters for hydraulic pulse train: WAGON parameters numbers of TRAIN WAGON and number of TRAIN SEASSON, manual calibration constants inputs (sub state: 22_State_Func)
4	3_State_Func	State_FLAGS_3_PULSE = ON AND ERROR = OFF	PULSE STATE, performing of hydraulic pulsation according of parameter settings, after finish of pulsing goes automatic to AUTO MODE (FLAGS_2_AUTO = ON, FLAGS_3_PULSE = OFF)
5	4_State_Func	State_FLAGS_4_ERROR	ERROR STATE, Safety relay determined, determination of the error source
6	5_State_Func	State_FLAGS_5	Sub state of PULSE STATE, measurement of the maximum of the hydraulic pulse peak during PULSE MODE (during the pause time and during the pulse time)
7	6_State_Func	State_FLAGS_21_QSET	Sub state of AUTO STATE, user setting of parameters for four TRAIN (T0, T1, COUNT, PRESSURE)
8	7_State_Func	State_FLAGS_7	Sub state of PULSE STATE, actual modification of parameters after one TRAIN SESSION

#### 4.5. Description of the Master Control State Machine

The main control state machine (Figure 4) for hydraulic pulse system consists of 7 separate threads running in the application layer, each thread represents the individual state of pneumatic–hydraulic pulsing system and is described in Table 3. Transitions between states in the application layer is controlled by the program flags and they are isolated from the hardware inputs signals (Table 1) processed by the physical layer (Table 2). This solution allows for the reliable function of high-pressure hydraulic pulse system, with reliable answers to user inputs or to the error status. We observed the reliable automatically generated ends of each hydraulic train pulsation cycle.

#### 4.6. Auxiliary Functions

The control of pulses sequences according to saved parameters in the hydraulic pulse train ensures thread 7\_State\_Func in the application layer, which is called from the state PULSE (3\_State\_Func). The thread 6\_State\_Func ensures the users inputs for controlling of the actual content of the hydraulic pulse train and is called from AUTO STATE (2\_State\_Func).

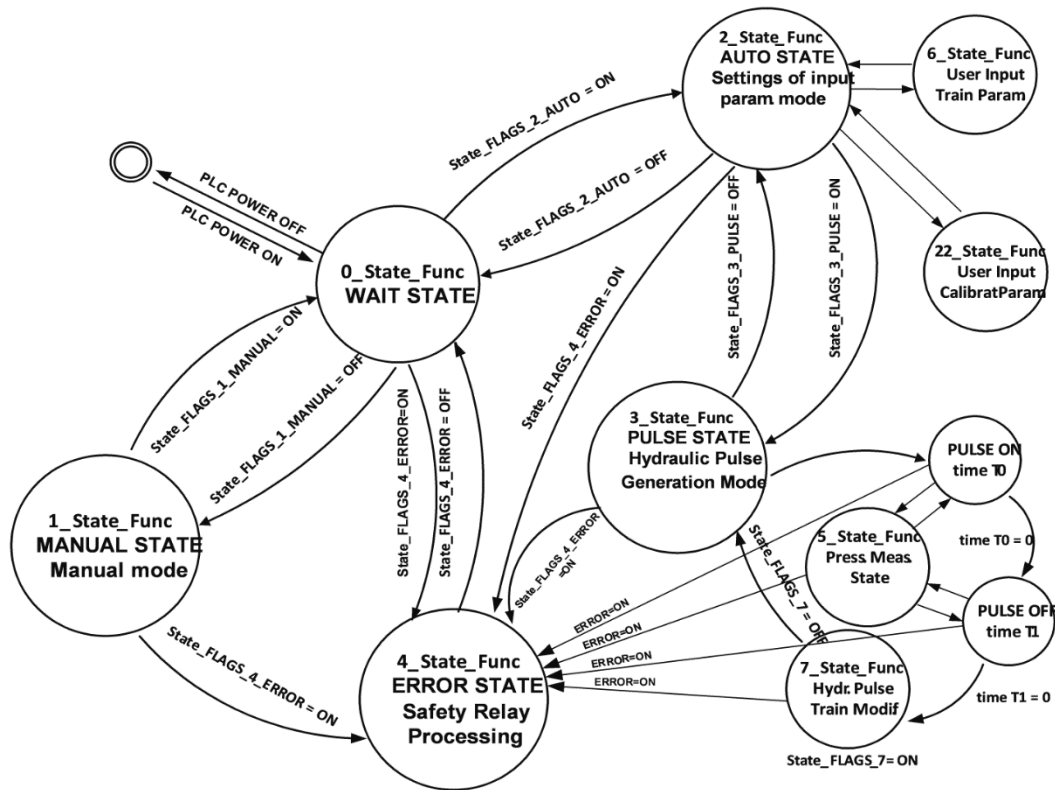
#### 4.7. Calibration of the System

The thread 6\_State\_Func in the application layer ensures the manual input of calibration constants from the user. The reason for the calibration can be described as follows: user works at input or at output with pressure units in [bar], but the PLC works with an internal representation of pressure in ADU units (12 bit). The user can perform the calibration process in MANUAL STATE which means static mode (no pulse condition, manually controlled output hydraulic pressure). The user obtains, during the calibration procedure, two calibration curves. The first calibration curve is dependence between the output hydraulic pressure (measured with portable calibrated hand-held hydraulic pressure measurement equipment from WIKA [10]) and the output voltage in 12bit ADU (measured via PLC itself) from the sensor output hydraulic pressure ATE PS60. ATE PS60 is a commercial produced hydraulic pressure sensor used in the cars, the manufacturer is ATE. In our case, in calibration process

we obtained a linear relationship, which leads to the linear Equation (1) derived by regression analysis from the graph of the measured values:

$$PWIKA [bar] = A_{ATE PS60} \times PATE PS60 [ADU] + B_{ATE PS60} \quad (1)$$

By the calibration-determined value of the constant  $A_{ATE PS60}$  for pressure sensor ATE PS60 is 0.1021 [bar/ADU] and the value of the constant  $B_{ATE PS60}$  is equal to 4.8074 [bar] (Figure 5).



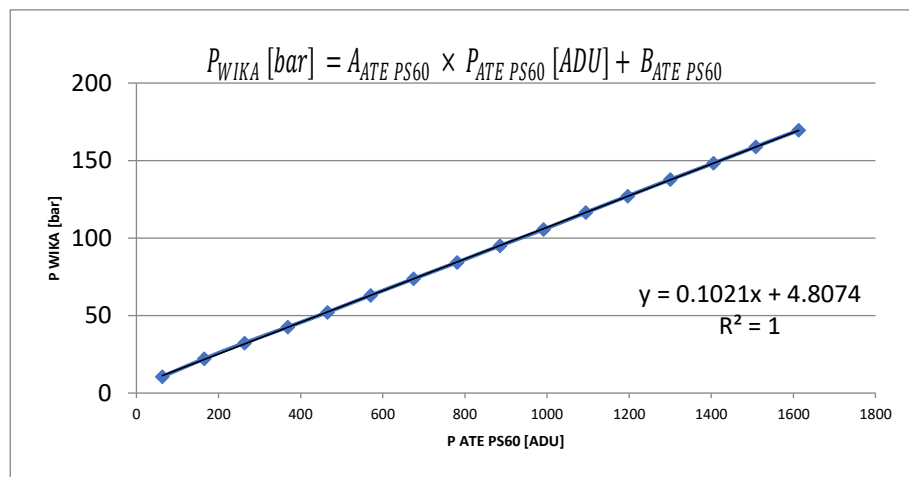
**Figure 4.** Master control state diagram for hydraulic pulse system consists of 7 separate threads running in the application layer, each thread represents the individual state of pneumatic–hydraulic pulsing system and is described in Table 3.

The second dependence is the static transfer function of the entire pneumatic–hydraulic system. This is the curve which represents output hydraulic pressure (in bar unit, measured by WIKA equipment [12]) at the values of the input control voltage for proportional pneumatic valve (measured in ADU units via PLC A/D extension module). Measurement leads to the next linear equation:

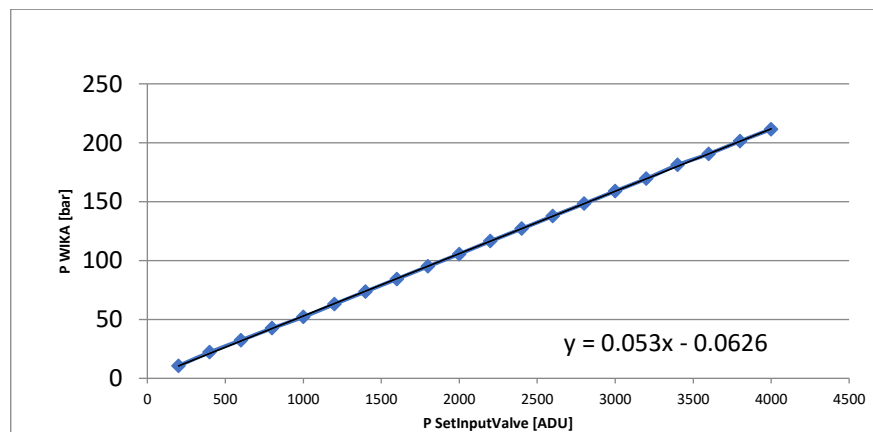
$$PWIKA [bar] = A_{SYS} \times PSetInputValve [ADU] + B_{SYS} \quad (2)$$

where the value of the constant  $A_{SYS}$  is 0.053 [bar/ADU] and the value of the constant  $B_{SYS}$  is 0.0626 [bar] (Figure 6). Both linear Equations (1) and (2) provide of four calibration constants, which user can enter in to the control system (in the thread 6\_State\_Func). Then, after the static calibration of pneumatic hydraulic pulse system, the user input entered value are in bar units obtained at the output of pneumatic–hydraulic system, as is required in both states—MANUAL and PULSE STATE.





**Figure 5.** Measured pressure system transfer curve for pressure sensor ATE PS60, dependence between pressure sensor ATE PS60 output voltage in ADU (Analog Digital Unit) units and pressure in bar from portable measurement device WIKA CPH 6200 (see Equation (1)).



**Figure 6.** Measured transfer curve between output of pneumatic-hydraulic system pressure in bar units and control voltage for proportional pneumatic valve as input of system in ADU units (see Equation (2)).

## 5. Discussion

The pneumatic-hydraulic system developed by us is one part in an industrial test bench. Our possible benefits are in the introduction of co-operating layers (physical layers and application layers) on a low-cost PLC platform (without the need for high-level programming like C++). This may be our possible contribution to the education of young technicians. All necessary software solutions (running at “nanoNavigator” IDE (Integrated Development Environment) from Phoenix Contact [13]) are provided by authors for study purposes as Supplementary Materials (Source code S1). Mutually communicating threads can be studied in the software simulator provided by nanoNavigator IDE and is not necessary to study or program with buy any PLC hardware.

## 6. Conclusions

We developed an embedded control for pneumatic-hydraulic system which produces a series of high-pressure hydraulic pulses (up 200 bar). The pneumatic-hydraulic system can be calibrated, it is networked in industrial networks (MODBUS over TCP/P and CAN bus) and generates accurately hydraulic pulses (time and pressures characteristics). The system proved to be reliable. This hydraulic impulse unit is a part of a test machine and it can be used in the industry for hydraulic parts dynamic testing by series of pulses with maximal frequency pulses 2 Hz. From the point of view of software

architecture, the embedded control system is based on a low-cost PLC architecture and control program is a set of parallel program threads. Each programming thread can be identified as one state in a master control state diagram. The PLC multi-thread program is not designed as one large and opaque linear program, but rather as a group of mutually communicating PLCs programs, which ultimately forms one master control state diagram. The cost of the described industrial networked control system with industry standard optoelectronic insulated interfaces and certified industrial safety relay does not exceed €1000 Euros.

**Supplementary Materials:** The following are available online at <http://www.mdpi.com/2411-9660/2/4/48/s1>, Source code S1: PLC software project (NanoNavigator IDE).

**Author Contributions:** Conceptualization, J.P.; Data curation, P.M.; Methodology, J.P. and P.M.; Software, J.P.; Validation, J.P. and P.M.; Writing – original draft, J.P.; Writing – review & editing, J.P.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Lamb, F. *PLC Hardware and Programming Multi-Platform*; AuthorHouse: New York, NY, USA, 2016; ISBN 13 978-1524648183.
2. Collins, K. *PLC Programming for Industrial Automation*; Exposure Publishing: New York, NY, USA, 2007; ISBN 13 978-1846854965.
3. Harel, D.; Lachover, H.; Naamad, A.; Pnueli, A.; Politi, M.; Sherman, R.; Shtull-Trauring, A.; Trakhtenbrot, M. STATEMATE: A working environment for the development of complex reactive systems. In Proceedings of the 11th International Conference on Software Engineering, Singapore, 11–15 April 1988.
4. Harel, D. A Visual Formalism for Complex Systems. 1987. Available online: <https://www.fceia.unr.edu.ar/asist/harel01.pdf> (accessed on 19 November 2018).
5. Merritt, H.E. *Hydraulic Control Systems*; John Wileys & Sons: Cincinnati, OH, USA, 1967.
6. Phoenix Contact. Nanoline Base Units. Available online: <https://www.phoenixcontact.com/online/portal/us/?uri=pxc-oc-itemdetail:pid=2700464&library=usen&pcck=P&tab=1&selectedCategory=ALL> (accessed on 19 November 2018).
7. Safety Relays-PSR-SCP-24UC/ESAM4/3X1/1X2/B-2900509. Available online: <https://www.phoenixcontact.com/online/portal/us?uri=pxc-oc-itemdetail:pid=2900509&library=usen&pdfmode=direct&pdflanguage=en> (accessed on 19 November 2018).
8. Communication Module-NLC-COM-ENET-MB1-2701124. Available online: <https://www.phoenixcontact.com/online/portal/us?uri=pxc-oc-itemdetail:pid=2701124&library=usen&pcck=P-21-03-03-01&tab=1> (accessed on 19 November 2018).
9. OLIMEX. AVR-CAN Development Board Based on AT90CAN128. Available online: <https://www.olimex.com/Products/AVR/Development/AVR-CAN/> (accessed on 19 November 2018).
10. John, K.-H.; Tiegelkamp, M. *IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids*; Springer: Berlin/Heidelberg, Germany, 2010; ISBN 978-3-642-12015-2.
11. Lewis, R.W. *Programming Industrial Control Systems Using IEC 1131-3*; The Institution of Electrical Engineers: New York, NY, USA, 1998; ISBN 0852969503.
12. Wika Instruments. Hand-Held Pressure Indicator Model CPH 6200. Available online: [http://www.wika.us/upload/DS\\_HPCPH6200\\_\\_\\_CPH6200\\_S2\\_en\\_us\\_17152.pdf](http://www.wika.us/upload/DS_HPCPH6200___CPH6200_S2_en_us_17152.pdf) (accessed on 19 November 2018).
13. Phoenix Contact. Software-NLC-NAV-01-2701221. Available online: <https://www.phoenixcontact.com/online/portal/us?uri=pxc-oc-itemdetail:pid=2701221&library=usen&tab=1> (accessed on 19 November 2018).



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).